

REMARKS

I. General

Claims 1-20 were pending in the present application. Claims 1-20 were previously rejected in an Office Action mailed July 13, 2005. In response, Applicant appealed the rejection to the Board, and the Board rendered its Decision April 10, 2007, reversing the rejection of claims 1-20. However, the Board entered a new ground of rejection as to claims 1 and 10. The new ground of rejection raised in the Board's Decision is:

- Claims 1 and 10 are rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,631,369 issued to Meyerzon et al. (hereinafter "*Meyerzon*").

In response to the Board's Decision, Applicant hereby requests that prosecution be reopened under 35 C.F.R. § 41.50(b), and further requests that the Examiner reconsider and withdraw the new grounds of rejection raised by the Board's Decision in view of the amendments and arguments presented herein.

II. Amendments

Claims 1, 10, 12, 14-17, and 19 are amended herein, and new claim 21 is added. No new matter is added by these amendments and newly added claim.

Claim 1 is amended to recite "receiving a request to process a URL in a desired manner". Claim 1 is also amended to recite processing the received URL "in said desired manner" in response to said received URL not matching any of said plurality of URLs. Claim 1 is also amended to recite "denying said request to said received URL in said desired manner in response to said received URL matching any of said plurality of URLs stored in said lexical search tree data structure." Support for these amendments can be found throughout the specification, *see e.g.*, page 4, line 30 – page 6, line 3 and FIGURE 2.

Claim 10 is amended to recite that the lexical search tree data structure stores a plurality of “hostile” URLs. Support for this amendment is found in the specification, *see e.g.*, page 5, lines 16-28.

Claims 12 and 14-16 are amended solely to properly reference the “hostile” URLs in view of the amendment to claim 10 from which they depend.

Claim 17 is similarly amended to recite that the lexical search tree data structure stores a plurality of “hostile” URLs. Claim 17 is further amended to recite “denying said processing of said received URL in response to said received URL matching said at least one hostile URL.” Support for these amendments can be found throughout the specification, *see e.g.*, page 4, line 30 – page 6, line 3 and FIGURE 2.

Claim 19 is amended solely to properly reference the “hostile” URLs in view of the amendment to claim 17 from which it depends.

New claim 21 is added herein. Claim 21 depends from claim 10 and further recites “said filter further operable to deny said processing of said received URL in response to said received URL matching any of said plurality of hostile URLs.” Support for this newly added claim can be found throughout the specification, *see e.g.*, page 4, line 30 – page 6, line 3 and FIGURE 2.

III. Rejections Under 35 U.S.C. §102

Claims 1 and 10 are rejected under 35 U.S.C. § 102(e) as being anticipated by *Meyerzon*. Applicant respectfully traverses these rejections below.

To anticipate a claim under 35 U.S.C. § 102, a single reference must teach every element of the claim, *see* M.P.E.P. § 2131. Thus, § 102 anticipation is not found when the applied art is lacking or missing a specific feature or the structure of the claimed invention. Further, the Federal Circuit has explained: “There must be no difference between the claimed invention and the reference disclosure, as viewed by a person of ordinary skill in the field of the invention.”

Scripps Clinic & Research Found. v. Genentech Inc., 927 F.2d 1565 (Fed. Cir. 1991). As discussed further below, claims 1 and 10, as amended herein, are not anticipated under § 102 by *Meyerzon* because *Meyerzon* fails to teach each and every element of these claims as required by M.P.E.P. § 2131.

Independent Claim 1

Claim 1, as amended herein, recites:

A method for Uniform Resource Locator (URL) filtering, comprising:
receiving a request to process a URL in a desired manner;
receiving an event notification upon the occurrence of an event associated with the received URL;
searching, in response to said event notification, a lexical search tree data structure storing a plurality of URLs for said received URL;
processing said received URL in said desired manner in response to said received URL not matching any of said plurality of URLs stored in said lexical search tree data structure; and
denying said request to said received URL in said desired manner in response to said received URL matching any of said plurality of URLs stored in said lexical search tree data structure.

Meyerzon fails to teach all elements of claim 1. For instance, *Meyerzon* fails to teach processing a received URL in a desired manner (e.g., as requested) in response to the requested URL not matching any of a plurality of URLs stored in a lexical search tree data structure, and denying the request to process the received URL in the desired manner in response to the received URL matching any of the plurality of URLs stored in the lexical search tree data structure.

Meyerzon is generally directed to a web crawler that crawls a computer network, identifies documents stored on the network, and creates an index for such documents. See Abstract of *Meyerzon*. “The primary application of the crawler is to build an index of a set of documents, so that the index can be searched by end-users that want to locate documents that match certain search criteria.” Col. 2, lines 26-29 of *Meyerzon*. The crawler creates a History Table containing a list of URLs for each folder and document found in the crawl, along with such information as local commit time (LCT) for each document and a deleted documents count

(DDC) and LCT or maximum LCT for each folder. *Abstract of Meyerzon*. The crawler then uses the associated information maintained in the History Table (e.g., LCT, DDC, maximum LCT, etc.) to determine how best to crawl the network to update the History Table and index over time. *See Abstract of Meyerzon*. Thus, *Meyerzon* is not concerned with determining whether to process a received URL request, but is rather merely concerned with crawling a network and populating an index and History Table with information identifying the documents stored on such network.

Meyerzon explains at column 2, lines 38-54 thereof:

Crawls typically are performed periodically to update the indexes with changed documents. Crawlers usually have no knowledge of the document store specifics. The only thing they can rely on is the last modified timestamp of the document, which is standard for most document stores, including HTTP servers, file servers, mail servers and databases. A problem with this approach is that, to ascertain the increment of the document set, the crawler must ask the corresponding server for each document whether the document's timestamp has changed. Since the percentage of documents that are unchanged between crawls is typically very high, it would be beneficial to minimize the number of requests the crawler makes to the document server to obtain the "increment" of the document set relative to the set of documents received during the previous crawl (i.e., to obtain new, modified and deleted documents). The present invention achieves this goal.

Meyerzon thus provides at column 2, line 64 – column 3, line 6:

This invention provides an improved mechanism for maintaining a document store in a manner that facilitates an efficient determination of whether and how the document store has been "incremented" or modified from a prior state. For example, the invention could be used in a Web crawler application, mail server, directory service, or any system requiring indexing or one-way replication of a document store. The invention is particularly directed to a method and system for identifying documents in a document store that have changed, are new, or have been deleted.

The Board's Decision concluded that *Meyerzon* discloses that its folders and documents in the History Table are arranged in a hierarchical tree format. Thus, the Board's Decision appears to have concluded that such a hierarchical tree of folders and documents in the History Table anticipates the recited lexical search tree data structure of claim 1. However, *Meyerzon*

fails to teach processing a received URL in a desired manner (e.g., as requested) in response to the requested URL not matching any of a plurality of URLs stored in a lexical search tree data structure, and denying the request to process the received URL in the desired manner in response to the received URL matching any of the plurality of URLs stored in the lexical search tree data structure. For instance, *Meyerzon* does not teach that a URL is processed in a desired, requested manner in response to the URL not matching any of the URLs stored to the History Table, and denying the request to process the received URL in the desired manner in response to the received URL matching any of the plurality of URLs stored in the History Table. In particular, *Meyerzon* does not address processing a URL in a desired manner (as requested) in the event the requested URL does not match a URL in the History Table, and denying such processing of the received URL in a desired manner (e.g., as requested) in the event that the requested URL matches a URL in the History Table. Instead, the History Table of *Meyerzon* is merely utilized to aid a crawler in determining how best to incrementally update its index of documents stored to the network.

In *Meyerzon* the History Table may be consulted by the crawler in determining whether to update its index and History Table. For example, *Meyerzon* explains at column 5, lines 21-38:

An incremental crawl retrieves only documents that may have changed since the previous crawl. The incremental crawl uses the index and History Table and its transaction log is seeded with the document address specifications (URLs) contained in the History Table.... To determine whether a substantive change has been made to the document, a Web crawler may filter extraneous data from the document (e.g., formatting information) and then compute a hash value for the remaining document data. The hash value would then be compared to a hash value stored in the History Table. Different hash values would indicate that the document has changed.

More specifically, *Meyerzon* explains its usage of the History Table at column 3, lines 20-48 as follows:

In an initial crawl, the crawler creates a first full index for the document store. The first full crawl is based on a set of predefined "seed" URLs and crawl restrictions, and involves recursively retrieving each folder/document directly or indirectly linked to the seed URLs. In the process of creating the first full index, the crawler creates a History Table containing a list of URLs for each folder and

document found in the first full crawl. The History Table also includes a LCT for each document and a DDC and LCT or MLCT for each folder. Flags are also included in the History Table to indicate which URLs have a corresponding DDC (i.e., which are folders) and which URLs have a parent with a corresponding DDC. Thereafter, in an incremental crawl, the crawler proceeds in accordance with the History Table (e.g., by starting at a first URL corresponding to a folder, as identified by the flag or bit mask, and continuing down through each folder URL) and determines (1) whether the DDC for that URL has changed and (2) whether the MLCT or LCT is more recent than the corresponding value in the History Table. If the DDC has changed, the crawler obtains a full list of items (URLs) in that folder, and compares the list with the URLs in the History Table to identify the deleted documents. The deleted documents are then deleted from the History Table and index. If the MLCT is more recent, the crawler queries the document store for the URLs of linked documents (i.e., linked to that folder) having a LCT more recent than the MLCT or LCT in the History Table for the folder. The History Table and index are then updated accordingly to reflect the changes to the document store.

Again, *Meyerzon* simply provides no teaching of processing a URL in a desired manner (as requested) in the event the requested URL does not match a URL in the History Table, and denying such processing of the received URL in a desired manner (e.g., as requested) in the event that the requested URL matches a URL in the History Table. Instead, the History Table of *Meyerzon* is merely utilized to aid a crawler in determining how best to incrementally update its index of documents stored to the network.

In view of the above, it appears that claim 1, as amended herein, is not anticipated by *Meyerzon* in the manner suggested by the Board's Decision because *Meyerzon* fails to teach all elements of claim 1. Therefore, the rejection raised by the Board's Decision regarding claim 1 should be withdrawn.

Independent Claim 10

Claim 10, as amended herein, recites:

A system for Uniform Resource Locator (URL) filtering, comprising:
a web server operable to receive a URL request from a client; and
a filter operable, upon receiving an event notification relating to said URL request from said web server, to search a lexical search tree data structure storing a plurality of hostile URLs, said filter further operable to process said received

URL in response to said received URL not matching any of said plurality of hostile URLs.

Meyerzon fails to teach all elements of claim 10. For instance, *Meyerzon* fails to teach a filter that is operable to search a lexical search tree data structure storing a plurality of hostile URLs and process a received URL in response to the received URL not matching any of the plurality of hostile URLs. Indeed, *Meyerzon* does not address hostile URLs at all.

As discussed above with claim 1, *Meyerzon* is generally directed to a web crawler that crawls a computer network, identifies documents stored on the network, and creates an index for such documents. *See* Abstract of *Meyerzon*. The Board's Decision concluded that *Meyerzon* discloses that its folders and documents in the History Table are arranged in a hierarchical tree format. Thus, the Board's Decision appears to have concluded that such a hierarchical tree of folders and documents in the History Table anticipates the recited lexical search tree data structure of claim 10. However, *Meyerzon* fails to teach that its History Table stores hostile URLs, and *Meyerzon* further fails to teach a filter that is operable to search its History Table and process a received URL in response to the received URL not matching any of the plurality of hostile URLs. Again, *Meyerzon* is not directed to filtering/denying processing of requested URLs that are hostile, but is rather merely directed to a crawler that identifies documents available on a network.

In view of the above, it appears that claim 10, as amended herein, is not anticipated by *Meyerzon* in the manner suggested by the Board's Decision because *Meyerzon* fails to teach all elements of claim 10. Therefore, the rejection raised by the Board's Decision regarding claim 10 should be withdrawn.

IV. New Claim 21

New claim 21 is added herein, and is believed to be allowable at least based on its dependency from claim 10 for the reasons discussed above.

IV. Conclusion

In view of the above, Applicant believes the pending application is in condition for allowance.

Applicant believes no fee is due with this response. However, if a fee is due, please charge our Deposit Account No. 08-2025, under Order No. 10017555-1 from which the undersigned is authorized to draw.

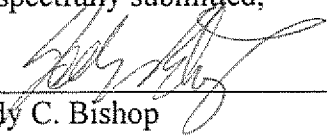
Dated: June 8, 2007

Respectfully submitted,

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being transmitted via the Office electronic filing system in accordance with § 1.6(a)(4).

Dated: June 8, 2007

Signature: Donna Forbit
(Donna Forbit)

By 
Jody C. Bishop
Registration No.: 44,034
Attorney for Applicant
(214) 855-8007
Fax (214) 855-8200